Adaptive cross-device videoconferencing solution for wireless networks based on QoS monitoring

Pedro Rodríguez, Álvaro Alonso, Joaquín Salvachúa, Enrique Barra, Javier Cerviño

Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid
Avda. Complutense 30, Ciudad Universitaria,
28040 Madrid, Spain
{prodriguez, aalonsog, jsalvachua, ebarra, jcervino}@dit.upm.es

***Abstract*—The increase in CPU power and screen quality of todays smartphones as well as the availability of high bandwidth wireless networks has enabled high quality mobile videoconferencing never seen before. However, adapting to the variety of devices and network conditions that come as a result is still not a trivial issue. In this paper, we present a multiple participant videoconferencing service that adapts to different kind of devices and access networks while providing an stable communication. By combining network quality detection and the use of a multipoint control unit for video mixing and transcoding, desktop, tablet and mobile clients can participate seamlessly. We also describe the cost in terms of bandwidth and CPU usage of this approach in a variety of scenarios.**

***Keywords*-multimedia, videoconference, wireless, mobile**

I. Introduction

Videoconferencing applications have been trying to successfully migrate to mobile devices for more than a decade. Traditionally, the low size and quality of the screens, available bandwidth and low processing power compared to desktop computers were really hard challenges that developers overcame with complex technical solutions that allowed users to communicate but often getting a much lower quality of experience.

Over the last years, the increase in popularity of smartphones has helped usher in a new era where high quality wireless communications are prominent. The more powerful CPUs and higher resolution screens of these devices allow for complex distributed applications that are similar in functionality to those found in desktop computers. This evolution has minimized and, in some cases, even eliminated the inherent challenges of mobile videoconferencing. Today, there is a wide variety of video and audio communications tools available for mobile platforms such as Skype, Google Hangouts and Apple's Facetime. However, at the time of writing this paper, none of these applications provided cross-platform videoconference with multiple participants displaying their videos at the same time. That is, those applications only display one video at a time in their mobile version.

We present a solution for the scenario of multiple participants communicating simultaneously via audio and video using different platforms. All users, regardless of their access device and network, have to be able to participate in equal conditions obtaining a similar experience that is adapted to their network and terminal limitations. We can divide the challenges tackled by this solution in two groups: device fragmentation and network conditions adaptation.

The variety of devices will be addressed by confining them into three categories: cell phones, tablets and desktop computers. Different applications with the same functionality are designed to work with each of the categories. A centralized server will provide videos encoded with different bitrates. Adapting to network conditions will be addressed by constantly monitoring the quality of service provided by the access medium. We designed an algorithm that decides how to react at any given time considering the input obtained from the network monitoring.

To sum up, in this paper we propose an architecture that enables an advanced videoconferencing scenario by providing cross-device video and audio communication tools for multiple participants. We have developed a demonstrator following this architecture as part of a research project called VaaS, undertaken between Universidad Politécnica de Madrid (UPM) and Telefónica. We have quantified the server and measured the resulting bandwidths in different scenarios.

II. Videoconference Challenges in Wireless Networks

A. Access Networks

Devices in different wireless networks can interact and have a videoconference together. So the videoconference system has to support all of them with the differences that they present.

The characteristics to be considered of these networks are: the bandwidth available at any given time to upload and download media from other participants, delay and jitter.

As we will explain throughout this paper, to overcome some of the challenges we used Flash technology that forced us to use TCP as the transport protocol. As seen in [1], high bandwidth communication coupled with delay and random packet loss such as the one encountered in wireless communications, significantly reduces the available throughput. Thus, it is necessary for a videoconferencing solution to be able to adapt to those random changes in throughput in order to be able to allow for fluent real-time communication. A

system that fails to adapt correctly will be imposing longer communication delays due to the packet retransmission present in TCP, and eventually the abrupt ending of the transmission.

B. Terminal capabilities

There is a huge variety of devices that can use a videoconference system, smartphones, tablets, desktop and laptops, even televisions are starting to incorporate this possibility with the apparition of the smart TVs.

But in comparison smartphones are the ones that have the more limited features when it comes to videoconferencing. So they are the ones that we give special importance in the scenarios and in the tests performed. The device features that limit a videoconferencing system are:

- CPU: the device will have to encode its video and audio before sending it and will have to decode one or several received videos and audios, this can be very CPU intensive.
- Battery: High CPU usage severly impacts battery life. In smartphones this also implies the heating of the device.
- Memory: the available RAM for the application is not very high in some devices.
- GUI: the screen size is very small in mobile devices compared to other devices such as tablets or desktop. This influences the layout, the way the videos are presented and the way the user interacts with the application. The objective is to provide the users with a similar experience on every platform.

III. VAAS

Our proposal to overcome the challenges listed above is called VaaS. VaaS is a centralized videoconferencing application with clients for smartphones, tablets and the web. It is based on Adobe Flash technology on the client side and Java on the server. VaaS provides videoconference *rooms* where up to six participants communicate with each other via audio and video.

The Adobe Flash platform allows easily portable applications via Adobe Air. With Air, a native application can be generated for those platforms that do not provide Flash support. This allowed us to create a single application for all mobile platforms with minor adjustments while, at the same time, being able to produce a web for desktop version without starting from scratch. Furthermore, we have previous successful experiences with the technology as seen in [2], [3] which also explain some of the basic technology used for this prototype in more detail.

The need to transcode videos to adapt to different terminals suggested a centralized server approach. This main drawback of this approach combined with Flash is that the only transport protocol available is TCP. At the time of developing this project, RTMP and its variants where the only way for a Flash application to transmit real time media to a central media server.

TABLE I
VIDEO RESOLUTION AND FRAMES PER SECOND

Feature	Upload Desktop.	Upload Smartphone.	LB Mix.	HQ Mix.
FPS	10	10	5	10
Resolution	320x240	320x240	80x60	320x240

A. Video transcoding and mixing

In a typical videoconferencing scenario, every participant sends his or her own video and receive one for each of the other participants. In our case, each client would send one video and receive five. That means that if all videos are of the same quality, the needed upload bandwidth equals the video bitrate while five times that throughput is needed in the download. This is acceptable when all participants have similar devices and network quality, as they will all get a good experience according to their capabilites. The problem can be even worse if participants choose their own quality as some could choose a very high resolution and quality video that others cannot possibly receive.

Furthermore, decoding several video streams at the same time requires significant CPU power. This is not a problem in desktop computers, however, even todays powerful smartphones have trouble coping with this workload and even if they are able to do it, the hit in battery life is very noticeable. That is why most videoconferencing applications in smartphones display only one video at any given time.

In order to tackle these problems, the central server will use two techniques widely known in real time communications scenarios: video mixing and transcoding.

We define video mixing in this context as combining several video streams into one. It can be done by taking frames from each of the videos and arrange them together in a new frame at the desired rate. Video transcoding is to adapt a video to a lower bitrate. By losing spatial and temporal resolution, the video can be transmitted using less bandwidth.

We combine these two techniques generating a video mosaic from all the participants and transcoding it to two bitrates that will be sent to clients depending on the conditions of their access network and the output of the algorithm.

Table I displays the chosen configuration for the prototype. Desktop and smartphone clients upload the same quality. Two mixed videos are generated, a high quality and a low bandwidth. As we will see in the next subsection, the adaptation algorithm will switch between the available qualities depending on the network situation.

Due to the solution being Flash-based the video and audio codecs and parameters are limited to those implemented in the Flash Platform. At the time of the development, only Sorenson Spark (a variant of H263) was available in Adobe Air for mobile operative systems. Speex is the codec used for audio.

B. QoS Monitoring and adaptation algorithm

This architecture also tackles the problem given in the mobile clients that are connected via wireless networks. The quality of these networks could vary depending on different

aspects: distance to the antenna, amount of clients connected at the same time, overall coverage of the service provider, etc. Furthermore, VaaS clients send and receive multimedia streams through TCP connections, this decreases the throughput when there is network congestion or the packets are sent through lossy transmission channels. In this section we will explain how we perform an active monitoring and adaptation of the traffic depending on the network condition.

Mobile VaaS clients automatically change the video quality of sending and receiving streams when they detect network problems. They can decrease the video quality when problems arise and increase it later when problems do not persist. This quality variation allows the system to use less bandwidth and avoid network congestion that is the most important source of problems in a TCP-based communication.

1) Initial QoS Measurement: The system obtains network parameters when a client connects to the server. During this phase the user will see a progress bar indicating that the system is measuring QoS parameters. These measurements calculate the latency and estimate the available bandwidth between client and server. Once finished the system configures the video to be sent from client to server to use half the bandwidth measured.

In the rest of the communication the system will follow a pessimist adaptive algorithm, assuming that there will never be better conditions than initial ones. If network congestion occurs during communication the system will take the initial measurements as a reference. Then, the system will try to reach again the initial state by dynamically increasing the video quality when no congestion is detected.

2) Network problem detection. Decreasing video quality: It will measure the time it takes to a packet travel from end to end ($latency$), and if this time increases above a threshold ($maxLatency$) the system will assume that the network is becoming congested. The system then reacts by decreasing the video quality in both sides of the communication, so that it consumes less bandwidth in both directions. We explain below the steps followed by the algorithm:

1) Every time the system detects network problems it reduces the bandwidth consumed by the client ($currentBW$) to half, decreasing the video quality (lines 2-4). If the consumed bandwidth is lower than a minimum limit ($minBW$) the client stops sending its video (lines 5-6).
2) If the connection does not improve the next step is to reduce the bandwidth consumed by the server in the same connection. In this case, the mobile clients that are subscribed to a mixed video could subscribe to another video with lower quality. This second video consumes less bandwidth (lines 7-9).
3) When problems persist, the clients will stop receiving video from the server. This case will be the same as they would be connected to a audio conference, because they do not send nor receive video streams (lines 10-11).
4) In the worst case, the problems persist. In this case the system will disconnect the client from the session (lines 12-13).

This algorithm will be executed by the system every $decRate$ seconds. This parameter should not be very low, because the system will decrease the quality very fast, and it is preferable to stabilize the communication on every change rather than detect false positives in the latency.

During this process the user will see different messages in the mobile application, indicating the different states of the connection.

3) Improved network conditions. Increasing video quality: The system assumes the network condition is favorable when it does not detect high latencies for a period of time. VaaS reacts trying to progressively increase the traffic rate up to its original state. In this case it follows a *generate and test* approach. In other words, the system increases the quality of the communication after a configurable time interval.

The system will increase the video quality step by step, in the opposite way it follows during the congestion state. In case network congestion is detected it will reduce the quality again. If the system does not detect congestion the upper quality limit is the one measured during the initial phase.

In this case this algorithm will be executed by the system every $incRate$ seconds.

We can see the relationship between the time to adapt to bad conditions and the time to adapt to good conditions as $k = incRate/decRate$. This parameter becomes a trade-off in our adaptive mechanism. Based on the practice k should be equal or higher than 1, because for the system it is more important not to experiment problems than to rapidly adapt to good conditions. But if this parameter is too high the system will react slowly to good network conditions.

IV. Demonstrator and Measurements

A. Experimental set-up

A total of six client devices are used in this scenario:

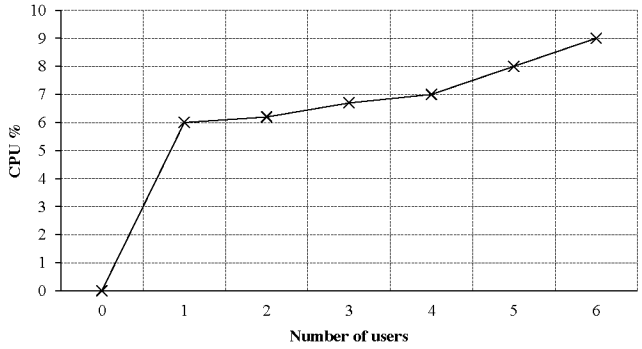
- Desktops: Intel Core 2 CPU, 2.40GHz and 4GB of RAM. The web application runs on Microsoft Windows and Google Chrome. We connected all desktops to the server through gigabit ethernet.
- Smartphones: Nexus S with 1GHz ARM Cortex A9 and 512MB of RAM. We used a native application that runs on Android 3.1, connected through wireless 3G network.
- Tablet: Samsung Galaxy Tab 10.1 with 1GHz Tegra 2 and 1GB of RAM. We used a native application that runs on Android 3.1, and we connected the tablet to the system through wireless LAN.

The server runs on a HP Compaq 600 Pro, which has an Intel Core 2 Quad CPU with 2.66GHz and 2GB of RAM.

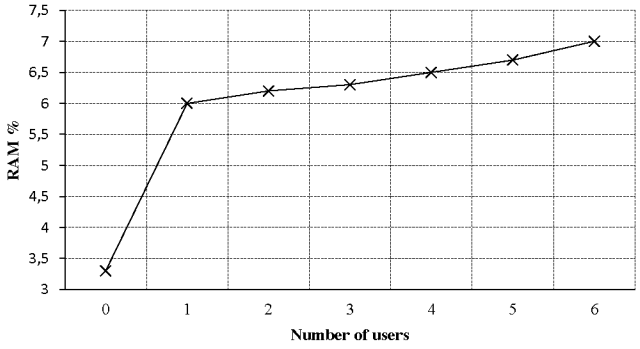
B. Server profiling

To characterize the server part of the system, we measured CPU and memory consumption. The percentages are taken from the total 2 gigabytes of RAM and of only one of the CPU cores.

We can estimate the total capacity for serving videoconference rooms in the server from the data obtained in these

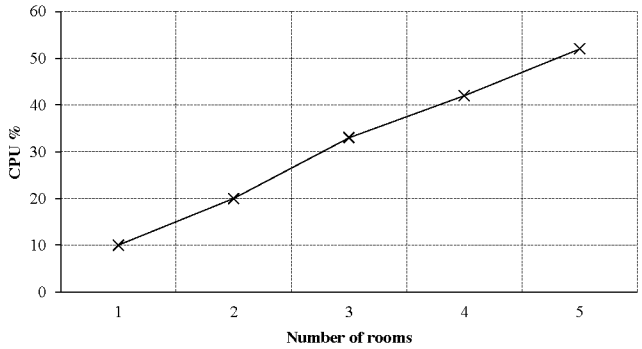


(a) CPU per user

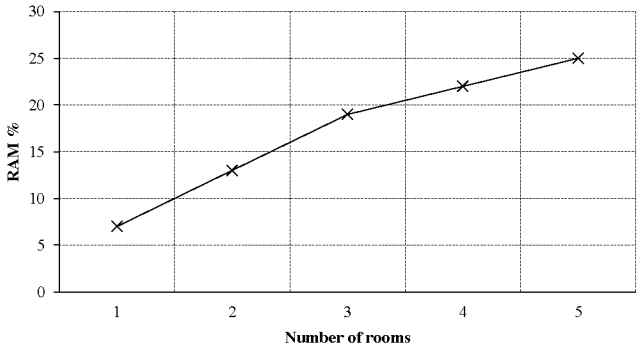


(b) RAM per user

Fig. 1. Resources per user in a single room



(a) CPU per full room



(b) Memory per full room

Fig. 2. Resources per full room

experiments. Fig. 1(a), 1(b) map the CPU and memory usage of a single room as the number of participants increases. The first user produces a steep jump in resource usage. This is expected as the transcoding is started and the necessary memory structures are allocated. However, as more clients enter the room, the stress over the CPU and memory increases almost linearly.

Fig. 2(a), 2(b) show the evolution of CPU and memory consumption with the number of rooms filled with six participants. As we can see, the increase is also linear. The CPU is at 60% when we reach five rooms but, as mentioned above, this measure is over only one core so there is plenty of room to grow. However, memory consumption is almost at a quarter of the total, meaning that, in this case, memory is the bottleneck in the server side.

C. Bandwidth measurements

In this experiment we want to analyse the output and input traffic in the server measuring the consumed bandwidth in a scenario with different types of devices connected to the system. These devices will be connecting gradually to the server. In the figures we can observe the captured input (Fig. 3) and output (Fig. 4) bandwidth in the server. We have measured the total aggregated bandwidth during the entire process and the aggregated bandwidth of each type of device since they are connected to the experiment.

We can obtain some conclusions from the graphics. First we can observe bandwidth consumption peaks when we connect each device. The peaks in seconds 20, 60 and 110 correspond to the connection of the desktop devices, the peaks in seconds 320 and 480 to the connection of the smartphones and the peak in second 610 to the bandwidth measure of the application in the tablet device.

Secondly, observing the total bandwidth capture between seconds 110 and 320 we can check the consumption when the three desktops are connected. Between seconds 320 and 480 the three desktops and a smartphone, between 480 and 610 the desktops and two smartphones and second 610 and the final the complete scenario with all the devices including the last smartphone and the tablet.

Thirdly, in the second line of Fig. 3 (desktop bandwidth) we can observe that there is not a perceptible variation of the consumed bandwidth when we connect the smartphones and the tablet to the session. However in the third line (smartphones) there is a little increase of the bandwidth when the tablet connects.

Finally we can see that the increase of the number of clients in the session does not affect to the upload bandwidth consumed by the rest of participants because of this only depends on the video stream than each client sends. Also the consumption of the desktops is greater than the smartphones.

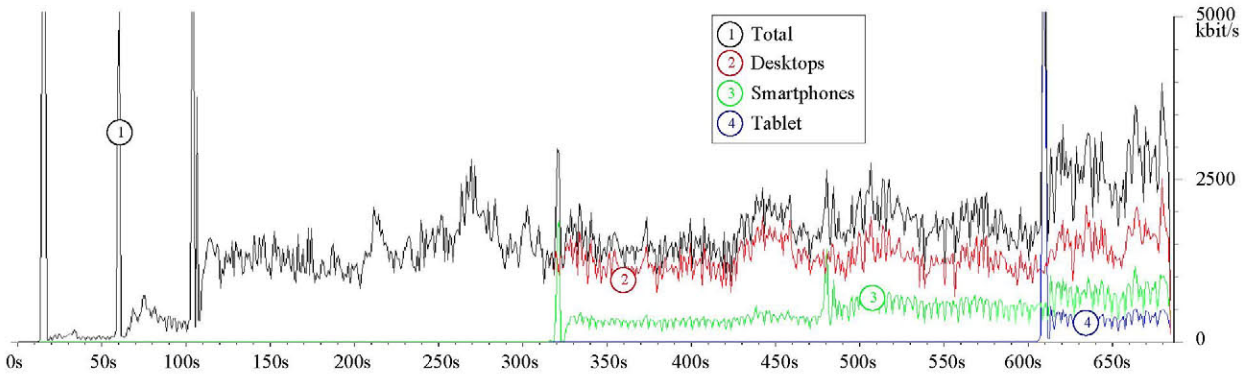


Fig. 3. Incoming Bandwidth

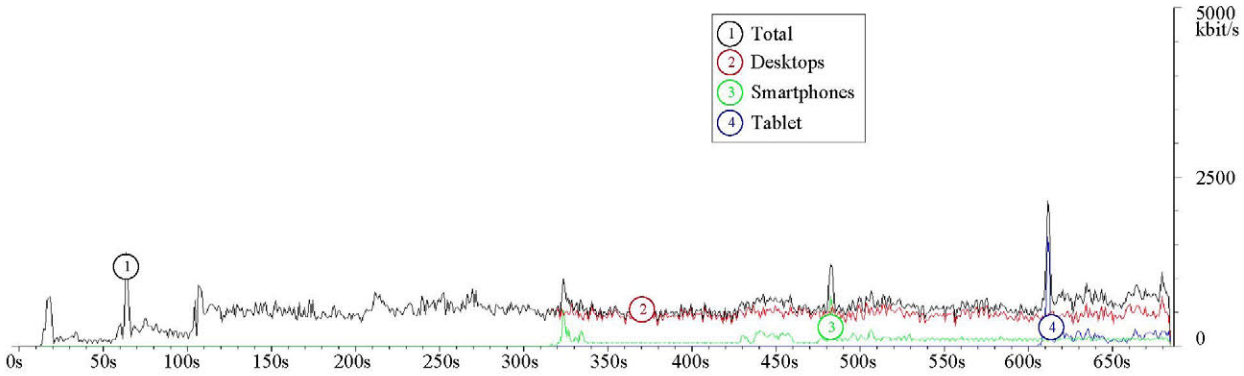


Fig. 4. Outgoing Bandwidth

V. Related Work

A. Network state management

We believe that the measurement of the network conditions must be done the simplest way possible in the sense of not adding complexity to the system. In [4] and [5] the proposal is to add a device near the mobile network that receives the multimedia traffic from the server and adapts it before redirecting it to the clients. Furthermore, this device checks periodically the network state in order to configure the adaptation.

To avoid including extra components in the system, [6] proposes sending signalling packets between clients and server indicating the state of the network and using this information to manage the quality configuration. In this solution no device is added to the environment but the definition of a new protocol is needed. Our solution implies the analysis of the network in the server based only in the traffic received.

A very good approach is proposed in [7]. In their solution each mobile device monitors some parameters like link quality info, signal strength at receiver, noise level at receiver and number of discarded packets due to different causes. Applying this parameters to different algorithms the device could adapt in different ways. As it is said in the study in most cases the client would make the adaptation keeping an acceptable frame rate.

In server-side, the system also manages the traffic quality adapting the media streams that it sends to the clients. The size of the frames received is directly related to the adaptation process, and this adaptation depends on the quality state. Therefore depending on the received frame size the server adapts the traffic sent to the client.

This way of estimating the mobile client state from the server can be imprecise depending of the video codecs used in the client or the different sizes of the emitted videos. However the detection is made using only the multimedia traffic without adding extra components or streams.

Solving all these problems, in our system the detection is made with the TCP latency monitoring. The server measures the latency between packets at the beginning of each connection and during the session. Depending on this latency it adapts

the video quality in both directions to the requirements of the network.

B. Video codecs

The authors in [8] explain how important the video transcoding is in a core network infrastructure to improve interactivity on mobile applications.

SVC (Scalable Video Coding) is explained in [9]. SVC is an extension of H.264/AVC, it codifies a high quality video stream that contains a subset of video streams. Using these subsets the codification can be adapted to the network conditions reducing the bandwidth. However, the Flash platform does not support scalable video codecs like the proposed by SVC standard. And, as explained in [10], even the H.264 codec implies many challenges when using in mobile environments with dynamical changes in the network conditions.

VI. Conclusions and future work

In this paper we have presented a cross-platform videoconferencing solution that aims to provide a consistent communication experience across different platforms, including smartphones. The use of mobile devices implies the use of wireless communications for a highly demanding real-time communication application as multiple participant videoconferencing.

The main challenges to overcome were the differences in terms of processing power and screen size of the terminals and the often-varying conditions of wireless networks.

Firstly, the fragmentation in the device space is overcome by designing the user interaction specifically for each platform while retaining the functionality. Adobe Flash, Adobe Flex and Adobe Air technologies were chosen to this end because of its multi platform nature and the flexibility of the tools.

Furthermore, transcoding and mixing videos in the server side helps smaller and less powerful devices to display several videos simultaneously without delay and improving battery life.

The use of the Flash platform imposes the use of TCP as a transport protocol when communicating with a central server. The use of a reliable protocol with retransmissions coupled with the varying conditions of wireless network can be problematic in real-time communication applications. We designed and implemented an algorithm that constantly monitors the network conditions and can affect the quality of the videos displayed in each terminal separately. Finally, we present the results of our measures in terms of bandwidth and CPU usage.

In the future, the work here can be continued in several ways. First of all, the use of Flash has proven to be effective when bringing communication to different platforms, however, its decrease in popularity and the arrival of HTML5 questions the viability of this solution in the short term.

WebRTC is being defined and developed to offer real-time peer-to-peer communications to the web taking advantage of HTML5 and existent real-time protocols and codecs instead of defining new ones. WebRTC is a joint effort by the WebRTC working group of the World Wide Web Consortium (W3C) and the rtcweb group from the Internet Engineering Task Force (IETF) where the first provide the HTML and JavaScript API and the latter defines the protocols and codecs to be used in the communication.

The first step towards implementing our solution in the WebRTC world would be to develop a MCU such as the one described in [11] capable of interconnecting users in a centralized way and being able to transcode and mix the received streams to adjust quality and bandwidth. Later, the QoS adaptation algorithm should be adjusted to the new conditions.

Acknowledgments

Authors would like to thank Telefónica for funding this project as well as the SAAN project (TIN2010-19138).

References

[1] T. Lakshman and U. Madhow, "The performance of tcp/ip for networks with high bandwidth-delay products and random loss," *Networking, IEEE/ACM Transactions on*, vol. 5, no. 3, pp. 336 –350, jun 1997.

[2] P. Rodriguez, D. Gallego, J. Cervino, F. Escribano, J. Quemada, and J. Salvachua, "Vaas: Videoconference as a service," in *Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009. 5th International Conference on*, nov. 2009.

[3] J. C. no Arriba, P. R. Pérez, J. S. Rodríguez, G. H. F. Toribio, and F. E. Cantero, "Marte 3.0: Una videoconferencia 2.0," in *Libro de Ponencias de la VII Jornadas de Ingeniería Telemática*, 2008, pp. 209– 216.

[4] A. Hokimoto and T. Nakajima, "Handling continuous media in mobile computing environment," in *6th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 1996, pp. 175–182.

[5] A. Hokimoto, K. Kurihara, and T. Nakajima, "An approach for constructing mobile applications using service proxies," in *Distributed Computing Systems, 1996., Proceedings of the 16th International Conference on*, may 1996, pp. 726–733.

[6] P. M. Ruiz and E. Garcia, "Adaptive multimedia applications to improve user-perceived qos in multihop wireless ad-hoc networks," in *in Multihop Wireless Ad-Hoc Networks, Proc. PIRMC 2002*, 2003, pp. 1467–1471.

[7] D. Esteban Ines, K. Fujikawa, E. Kawai, and H. Sunahara, "Streaming mobile multimedia optimization for video-conferencing scenarios," in *Advances in Multimedia, 2009. MMEDIA '09. First International Conference on*, july 2009, pp. 80–85.

[8] B. Shen, W.-T. Tan, and F. Huve, "Dynamic video transcoding in mobile environments," *MultiMedia, IEEE*, vol. 15, no. 1, pp. 42–51, jan.-march 2008.

[9] T. Schierl, T. Stockhammer, and T. Wiegand, "Mobile video transmission using scalable video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1204 –1217, sept 2007.

[10] Y.-M. Hsiao, J.-F. Lee, J.-S. Chen, and Y.-S. Chu, "H.264 video transmissions over wireless networks: Challenges and solutions," *Computer Communications*, vol. 34, no. 14, pp. 1661 – 1672, 2011.

[11] P. Rodríguez, J. Cerviño, I. Trajkovska, and J. Salvachúa, "Advanced topics in videoconferencing based on webrtc," in *2012 IADIS Collaborative Technologies*. Lisbon, Portugal: IADIS, 2012.